Efficient Physics-Based Implementation for Realistic Hand-Object Interaction in Virtual Reality

Markus Höll[1]* Markus Oberweger[1] Clemens Arth[1] Vincent Lepetit[1,2]
[1]Institute of Computer Graphics and Vision, Graz University of Technology, Austria
[2]Laboratoire Bordelais de Recherche en Informatique, Université de Bordeaux, France

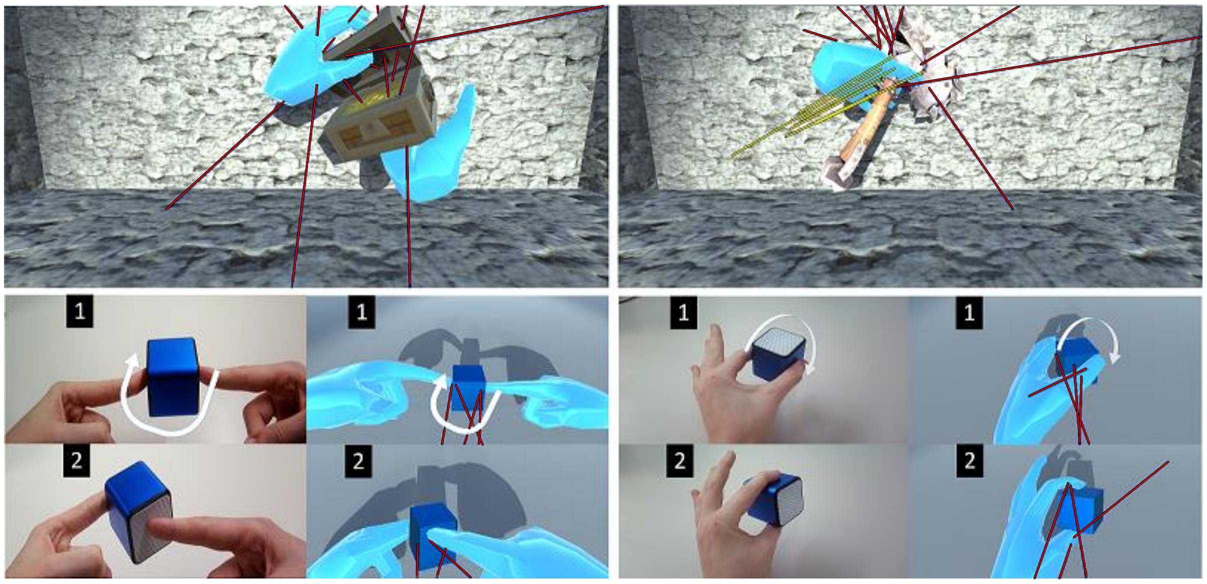


Figure 1: Demonstration of our physics-based hand interaction method for VR environments. Our approach allows countless complex interactions such as handling a box in one hand and opening it with the other (top-left), or smashing a wall with an axe held in one hand (top-right). We can even spin an object between two fingers from two different hands (bottom-left) or from the same hand (bottom-right) in a very realistic manner. The red arrows indicate the forces calculated with our method and applied to the objects. Yellow arrows depict the global hand motion depicted for demonstration purposes only.

Abstract

We propose an efficient physics-based method for dexterous 'real hand'-'virtual object' interaction in Virtual Reality environments. Our method is based on the Coulomb friction model, and we show how to efficiently implement it in a commodity VR engine for real-time performance. This model enables very convincing simulations of many types of actions such as pushing, pulling, grasping, or even dexterous manipulations such as spinning objects between fingers without restrictions on the objects' shapes or hand poses. Because it is an analytic model, we do not require any prerecorded data, in contrast to previous methods. For the evaluation of our method, we conduction a pilot study that shows that our method is perceived more realistic and natural, and allows for more diverse interactions. Further, we evaluate the computational complexity of our method to show real-time performance in VR environments.

Index Terms: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically-based Modeling; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Direct Manipulation

*E-mail: {hoell,oberweger,arth,lepetit}@icg.tugraz.at

1 Introduction

Virtual Reality but also Augmented Reality applications require interaction between the user and the virtual elements. Controllers such as the Nintendo Wii-Mote [44], Oculus Touch, HTC Vive Controller, or HoloLens Clicker can be used. However, they are cumbersome and not very intuitive to use, and the possibility of using our bare hands for the interaction is very appealing for VR games and VR environments. It can even play a key role in medical applications [9].

Many methods and hardware [14, 28, 51] have been developed to accurately capture the 3D pose of the user's hands. This is, however, not sufficient for realistic hand interactions: If the user's hand is simply modeled as a 'kinematic object', the virtual hand follows the tracking information for the real hand motion and can penetrate the virtual objects, which can lead to very unstable results in physics engines [16]. The Leap Motion Interaction Engine[1] does solve this penetration problem and supports grasping but in a non-physics-based way as the hand motions are directly transferred to the held objects.

To be convincing, the physical interaction between the hands and the virtual objects needs to be realistically simulated. Because such simulation can be very complex, most proposed interaction approaches are simplified, constrained, or artificial. For example, some methods recognize gestures that trigger some predefined interactions [7, 25, 45]. Other methods require some prior training phase [34, 37]. Physics-based methods, like ours, provide more

[1]https://developer.leapmotion.com/unity/

unconstrained interaction possibilities [4,16,17,27]. However, currently, these methods are mostly limited to simple grasping interactions.

In the real world, our ability to interact with objects is due to the presence of friction between the surfaces of the objects and our hands, as friction is a force that resists motion. We therefore aimed to take into account friction forces correctly in our approach. We propose to use the Coulomb friction model for properly simulating the forces the user applies to the objects according to the 3D pose and motion of his/her hand, to counteract external forces such as gravity or collision forces. The Coulomb friction model captures under the same framework both static and dynamic effects that happen when two objects are in contact. It is only an approximation of real dynamics, but as we will show, it produces convincing motions while remaining tractable, in contrast to approaches such as [39,41], which rely on a more complex physics model but are much less efficient and thus not applicable for real-time VR. In other words, we argue that the Coulomb friction model is a good trade-off between realism and tractability for interaction in VR.

This model choice indeed allows us to simulate interactions much more complex than simple grasping. For example, with our approach, the user can push, pull, let an object slide along his/her hand if (s)he does not grasp it firmly enough, or even spin an object between two fingers. Fig. 1 illustrates various types of interactions possible with our approach.

Our method starts from the 3D hand pose—in practice we use the Leap Motion device to estimate it. Using a simplified 3D model of the hand, we detect the intersections between this hand and the virtual objects. From these intersections, we define contact points between the virtual object and a virtual hand model, which corresponds to the real hand just before it starts penetrating the objects. We then show how to apply the Coulomb friction model to these contact points, taking into account the force applied by the user, which we take proportional to how much his/her real hand penetrates the virtual objects, and the friction parameters of the materials of the objects to compute the final resulting forces. We can then introduce them in a physics engine such as PhysX, which simulates the objects' motions.

In the remainder of the paper, we first give a short overview over related works. In Section 3, we show how to estimate the contact points, how to update contact points during hand-object interaction. and how to calculate the forces that are applied to the virtual object for the physics simulation. Then, in Section 4 we show qualitative results of our method applied to several interaction tasks. In Section 5 we present the results of a pilot study and a performance evaluation.

2 Related Work

3D interaction with virtual objects has recently become an active research field in Human Computer Interaction and Computer Graphics, with the introduction of 3D interaction systems [15,31,38,42] and vast improvements of VR technologies over the last couple of years [1,3,4,16,17,20,21,27,33,45,50].

Recent kinematic skeleton tracking sensors such as Microsoft Kinect or Leap Motion made research on real-time human interaction systems much more accessible. The Kinect sensor enables easier hand-gesture and human-activity recognition [49]. Empirical work [14] has shown that the Leap Motion is a reliable and accurate system for hand skeleton tracking, which recently got improved with a new Orion tracking software[2], specifically developed for VR environments. While these two sensors provide good results for hand tracking, research has since focused on hand interaction methods. We structure these methods in four different categories to give a more detailed overview of hand-based 3D interaction systems within virtual environments.

[2]https://developer.leapmotion.com/orion/

Kinematic Gesture-Based Approaches.

A frequent and simple type of hand interaction methods relies on kinematic- and gesture-based interfaces, where a predefined set of gestures is used to perform certain assigned actions. Common gestures include circle, swipe, pinch, screen tap, and key tap gestures [6,7,45]. A pinch detection is often used for detecting intended grasp interactions [25]. One popular example of such a gesture-based interface is the Microsoft HoloLens [1,45]. A few predefined hand gestures are detected in real-time to trigger certain interactions with virtual objects, similar to mouse clicks. Leap Motion recently released Interaction Engine[3], which also makes use of kinematic grasps to perform simple object transformations. Due to the constrained action space, this approach is also appealing for Machine Learning, where [26,36] for example detect hand gestures for applications such as free hand control of automotive interfaces. While these interaction approaches makes sense for certain VR environments, the interaction is artificial, highly limited in their possibilities, and thus not sufficient for direct object manipulation.

Heuristic-Based Approaches.

These approaches require predefined heuristics or *a priori* information about the hand and/or object to perform a limited set of object interactions, mostly object grasping [22,24,34,37]. A data-driven grasp method needs to synthesize prerecorded, real hand data to identify the most similar one during runtime from a predefined database [22,24,37]. Object interaction is only possible with object shapes [24,37] or object categories [22] that are predefined. These requirements induce significant drawbacks, including the need for prior information of either hand and finger poses and/or object shapes, and limitations given by interaction restrictions when synthesizing real-time hand poses from the prerecorded grasp database. This significantly limits the practical application of such methods in unconstrained environments.

Physics-Based Approaches.

Entirely physics-based hand interaction systems were recently not widely researched due to the complexity and challenges, such as speed and stability of the physics simulation [10], or accuracy of hand tracking [40]. Hilliges *et al.* [16] showed with HoloDesk an entirely physics-based interaction system using particles on the hand mesh that induce forces for object grasping. The method suffers from occlusion problems due to the tracking hardware especially when grasps are performed. A finger-based deformable soft body approach on convex objects using soft pads on rigid finger bones has been proposed by Jacobs and Froehlich [17]. Limitations are due to the missing palm enabling only finger-based interactions and powerful grasps can cause the soft pads to collapse. In order to increase realism of the interaction, more complex friction models can be used, such as the *Coulomb-Contensou* model [39]. Despite the accurate results, the computational complexity of this model is very high, which limits its applicability in VR where real-time performance is required. A more realistic physical simulation of the hand deformations can also consider the hand flesh [12,32], however, these approaches are computationally expensive and themselves require approximations. Another unconstrained grasping method has been proposed by Borst and Indugula [4] who use a spring-damper model and haptic feedback gloves. This approach has limitations due to the reduced degree-of-freedom per fingertip caused by the gloves. Also, the exerted forces are only applied with respect to the fingertips, which limits the grasping possibilities. Another physics-based grasping algorithm introduced by Nasim and Kim [27] uses a second dynamic proxy hand. However, since the physics forces are applied through the proxy hand that gets frozen after a contact occurred, the amount of interaction possibilities besides grasping and pushing are not clear and rather limited.

[3]https://developer.leapmotion.com/unity/

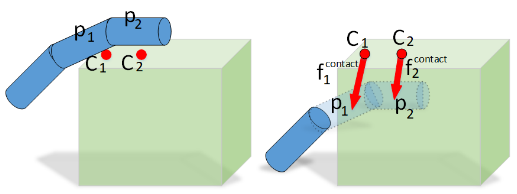


Figure 2: Estimating the contact force. The contact force $\mathbf{f}_i^{\text{contact}}$ is proportional to the distance from the contact point $\mathbf{C}_i$ to the current location of the intersecting phalanx $\mathbf{p}_j$.

Hybrid Approaches. We consider several methods as a hybrid category since they use certain aspects from different hand interaction categories and combine them together. Recent approaches compute initial contact forces via small particles sampled across the virtual skin mesh [20, 21]. However, as soon as an a priori defined force threshold is reached, the virtual object is considered as grasped and simply set to kinematic, thus following the virtual hand with global translation and rotation. Other approaches use a mixture of previously recorded grasps and synthesize new grasps with a computer graphics physics-driven virtual hand [33, 50]. Since these methods rely on prior knowledge or a database of grasps, interactions are still limited.

Comparing our physics-based hand interaction system to these related works, our method is entirely physics-based, fully unconstrained in its nature, and very efficient enabling integration into computational expensive VR scenes. Besides having such an unconstrained property, our algorithm also supports stable motion controls similar to methods that focus primarily on grasping [50]. Therefore, we explicitly model exerted contact forces on contact points of the virtual phalanges. We do not rely on any predefined data, states, or conditions, which allows us to perform unconstrained hand and finger interactions known from the real world with arbitrary virtual objects. Our method also allows for finger dexterity and thus unconstrained and dexterous 3D object manipulation during interaction.

3 Physics-Based Hand-Object Interaction

In this section we give a detailed description of our algorithm and explain the underlying physical principles that we use for our approach. First, we show how to identify interactions between the physical user's hand and the virtual objects, and contact points between them. Using these contact points, we calculate the forces that are induced by the hand interaction and apply them to the virtual object. We further show how to implement them efficiently into an existing physics engine.

3.1 Phenomena of Friction

Friction is a very complex phenomena. It models the tangential reaction force between two surfaces in contact. Several models have been introduced to simulate friction as close to real friction as possible. In reality, friction depends on many components such as the surface materials, temperature, wear, the topology of the objects, relative velocity, or the presence of lubrication.

The *Coulomb Friction Model* is by far the most well-known model to approximate the friction phenomena. It is a rather simple yet good approximation. For other models to approximate friction, we can refer to [8, 11, 19, 29], for example. However, since our contribution relies in realistic hand interaction method for real-time VR, we show that the Coulomb model presents a good trade-off between accuracy and computational efficiency.

3.2 Hand-Object Contact Points

In order to compute the forces which are applied by the hands to the virtual objects, we first need to identify the *contact points* between the user's real hand and the virtual objects. As we mentioned in the introduction, hand-object interpenetration can become a significant problem in physics engines, and we explain here how we efficiently solve the interpenetration problem when estimating accurate contact points between the virtual hand model and the virtual objects.

A contact point is defined as a 3D point that lies both on the hand surface and the surface of the virtual object. To identify such a point, we rely on the simplified hand model, which is made of simple shapes: Three capsules per finger and one cuboid for the palm. Each of these shapes can have one contact with the object. In practice, we continuously track the 3D pose of this hand model in terms of 3D location and orientation of each phalanx using the Leap Motion device.

As the real hand can penetrate the virtual objects, it is not clear how to define meaningful contact points. To do so, we propose the following method: We continuously look for potential collision by defining a small threshold distance around the virtual objects. We create a contact point when a point on the hand model's surface gets closer to a virtual object than this threshold. This event can be detected efficiently using a physics engine. Once we detected a point on the hand that is closer to the object surface than the threshold, we take the closest 3D point on the object surface to this point as the contact point, denoted $\mathbf{C}_i$. We will use this point to compute and apply the different forces sampled within a small area around this point as explained below.

In practice, the human skin is a soft body and forms a contact area rather than a single contact point [13], which helps to stabilize exerted forces when being in direct contact and keeps control over the object manipulation. We approximate this property by adding multiple contact points sampled on the surfaces of the hand and of the object around the initial contact point. Contacts on the thumb and the palm are sampled over a wider area than the other four fingers.

Updating the contact points during object interaction represents a significant challenge due to the interpenetrating property of the real hand and virtual objects. Updating the contact points is necessary when a contact is slipping along the surface of an object. A well known method to update contact points on the object's surface is the *God-Object* approach [18, 30]. Here we use a simple ray-based technique to obtain the contact points as it is sufficient for our purpose. Such methods can be found within the field of haptic rendering [2, 43] that requires very efficient algorithms.

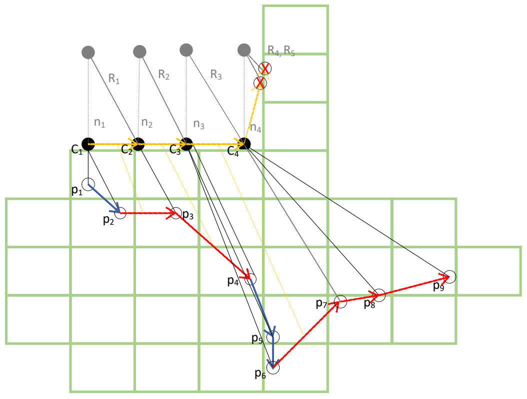


Figure 3: Updating the contact points along the object's surface using raycasting. The image illustrates the trajectory of one phalanx inside a 3D voxelized virtual object from a top-down view. Blue arrows depict static (stable) motions, red arrows dynamic (slippy) motions, and $\mathbf{R}_n$ illustrates rays in direction of $\mathbf{p}_j$.

As we explain below in Section 3.3, we distinguish between sliding (dynamic) and stable (static) contact states. As long as a contact remains in the stable state, there is no need to update the contact point since the contact point is not moving. However, when we detect a sliding contact state, we estimate a new contact point on the objects surface. We illustrate our solution in Fig. 3. For more details see Section 3.5.

3.3 Computing the Contact Force

Our method considers the normal- and tangential component of the contact force. The contact force is computed for every phalanx in contact with an object and applied to the corresponding contact area. As we will explain, the Coulomb friction model computes the tangential friction forces from the tangential component of these forces, to take into account the frictions along the surfaces. The Coulomb friction model distinguishes between two types of friction: dynamic and static friction. Static friction happens on stable contact points and has usually a higher friction coefficient, whereas dynamic friction occurs when objects are in relative motion, that is, when an object slides along the surface of the hand. The criterion to distinguish between static and dynamic friction forces is based on a 'friction cone', as explained below.

Contact Force. We define the contact force $\mathbf{f}_i^{\text{contact}}$ as:

$$\mathbf{f}_i^{\text{contact}} = \gamma(\mathbf{C}_i - \mathbf{p}_j)\,, \tag{1}$$

where $\mathbf{C}_i$ denotes the contact point on the object's surface, and $\mathbf{p}_j$ is the 3D centroid of the hand phalanx, which is tracked by the Leap Motion sensor. This formulation resembles a simple spring model [5]. When the user interacts with an object, $\mathbf{p}_j$ intersects the object's volume since the hand model follows the pose estimated for the user's real hand. This formulation is illustrated in Fig. 2. For the experiments in this paper, we set $\gamma = 100$. We also investigated a way to automatically adjust γ with respect to the physical properties of the object (volume, mass, surface material) to ensure that the user can bring up enough force even for small or heavy objects. We refer to the supplemental material for a more detailed discussion.

With the expression of Eq. (1), the direction of the contact force is dynamically updated according to the phalanx's current position, thus allowing the user to control the force finely.

The normal component of the contact force is calculated as:

$$\mathbf{f}_i^{n\text{-contact}} = (\mathbf{f}_i^{\text{contact}} \cdot \mathbf{n}_i)\mathbf{n}_i\,, \tag{2}$$

where $\mathbf{n}_i$ is the surface normal vector of the object mesh at the contact point $\mathbf{C}_i$. The normal component of the contact force is applied directly to the object. The Coulomb friction model computes the force $\mathbf{f}_i^{T\text{-contact}}$ applied in the tangential direction from the tangential component of the contact force to take into account the frictions along the object surface.

The expression of the tangential component of the contact force is:

$$\mathbf{f}_i^{t\text{-contact}} = \mathbf{f}_i^{\text{contact}} - \mathbf{f}_i^{n\text{-contact}}\,. \tag{3}$$

The expression of the tangential force $\mathbf{f}_i^{T\text{-contact}}$ depends whether or not the contact force $\mathbf{f}_i^{\text{contact}}$ is inside the *friction cone*. The friction cone is defined by the Coulomb friction model as a cone with the vertex corresponding to the contact point and the axis along the surface normal $\mathbf{n}_i$. The contact force $\mathbf{f}_i^{\text{contact}}$ is inside the friction cone if and only if the following condition is true:

$$F_i^{\text{inside}} = \mathbf{f}_i^{\text{contact}} \cdot \mathbf{n}_i > 0 \quad \wedge \quad \|\mathbf{f}_i^{t\text{-contact}}\| \leq \mu_i^{st}(\mathbf{f}_i^{\text{contact}} \cdot \mathbf{n}_i)\,, \tag{4}$$

where μ_i^{st} is the static friction coefficient at the surface location of the contact point $\mathbf{C}_i$.

The tangential force $\mathbf{f}_i^{T\text{-contact}}$ can then finally be computed as:

$$\mathbf{f}_i^{T\text{-contact}} = \begin{cases} \mathbf{f}_i^{t\text{-contact}} & \text{if} \quad F_i^{\text{inside}} \quad \text{is true} \\ \mu_i^{dyn} \cdot \mathbf{f}_i^{t\text{-contact}} & \text{otherwise.} \end{cases} \tag{5}$$

This force lets the object slide along the hand surface or on the contrary lets the hand grasp firmly by counteracting gravity, depending on its magnitude and direction, while taking into account the friction properties of this surface. We finally apply the forces $\mathbf{f}_i^{T\text{-contact}}$ and $\mathbf{f}_i^{n\text{-contact}}$ at contact point $\mathbf{C}_i$. For the material-dependent friction coefficients μ^{st} and μ^{dyn}, we use values from experimental data [48].

Tangential Friction Force The force from the tangential part of the contact force transfers the tangential motion of the hand onto the object in the direction of the hand movement. This force allows the hand to actually move the object. External forces such as gravity or collision forces can also influence the tangential component.

Fig. 4 illustrates the importance of the tangential component that allows to firmly pull an object with the hand motion. The hand moves for a small distance within the time interval Δt. This movement induces an increase of the tangential component of the contact force since the phalanges are moving away in tangential direction from the contact points. If the increase of the tangential component within the time interval Δt is too large in relation to the normal component, the contact switches to the dynamic state and the contact point starts to slip on the object's surface. The same principle can be observed under the influence of external forces such as gravity or collision forces. Fig. 6 illustrates gravitational acceleration during dynamic contact states which induce an increase of the tangential portion and thus cause the object to slip between the phalanges since the contact was not stable. Depending on the contact direction of each contact force, the global hand motion can increase the tangential or normal component of each friction cone and consequentially may contribute to its normal part.

We show a 2D projection of a friction cone in Fig. 5. The left figure shows the friction cone for the case when the hand motion is in the same direction as the initial contact direction of a phalanx. This consequentially increases the normal part of the resulting force allowing for larger tangential forces according to the Coulomb friction model, and the friction cone enlarges from the dashed line to the solid line. The same applies for collision forces as external forces which would occur in opposite direction of exerted contact forces.

The right figure of Fig. 5 shows the case when the global hand motion is in strong tangential direction to the surface. The new resulting contact force shows an increase of the tangential component and does not contribute to the normal component, thus it does not change the friction cone, but it increases the tangential component that is applied to the object and decreases the maximum of the allowed tangential force according to the second expression of Eq. 4. Again, the same principle occurs as a consequence of external forces. If the gravitational acceleration of an object is not counterbalanced by sufficient contact force, an object would slip between fingers during a grasp interaction increasing the tangential component of the dynamic contact state.

3.4 Visual Representation of the Hand

For the visual representation of the hand, we use a semi-opaque stylized rigged hand mesh. As pointed out by Nasim and Kim [27], a semi-opaque property helps users to receive a better depth perception and thus supports interaction planning. We have noticed that a stylized hand model has the benefit of avoiding uncomfortable user experiences, which can happen with too realistic hand models [47].

Since our tracked hand is able to interpenetrate the virtual object, we render an opaque hand mesh [35] with the pose from the moment an interaction occurred. To avoid visual interpenetration, we freeze

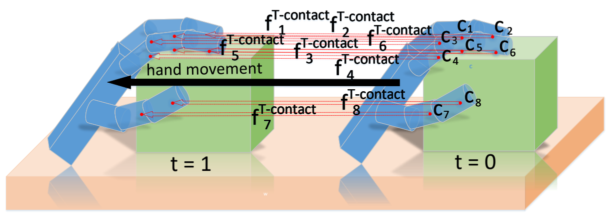


Figure 4: Illustration of an increased tangential force $f_i^{T-contact}$ from $t=0$ over $t=1$ causing an object to move in the the direction of the global hand motion during contact.

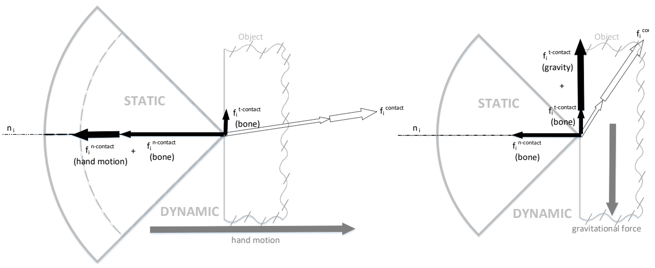


Figure 5: Left: contribution of the global hand motion to the normal portion of the established contact. Right: increasing tangential force due to the slipping state caused by gravitational acceleration.

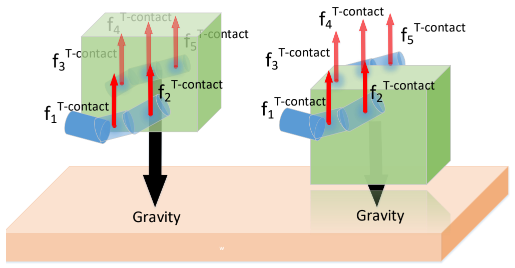


Figure 6: An example of increased tangential portion by gravitational acceleration as an example of external force. The black arrow depicts the gravity, the red arrows illustrates the tangential friction force to counteract gravitational acceleration during dynamic contact state.

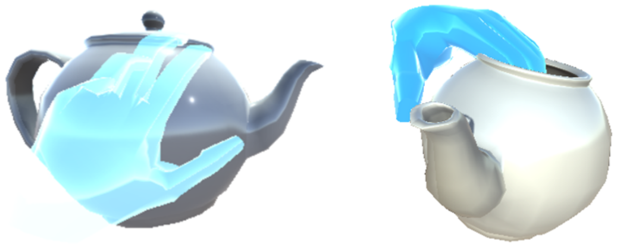


Figure 7: Left: Semi-transparent hand mesh rendered before contact. Right: Opaque hand mesh rendered during contact.

the hand mesh and update its rigid position by applying the relative global transformation of the virtual object. The hand mesh can still be slightly updated and refined to some degree once other fingers interact with the object. This ensures that the user can see a reasonable rendered hand pose of its interaction which is not penetrating or causing any confusing visuals. As shown in Fig. 7, this should help the user to constantly have a reasonable render of the interacting hand pose and avoid visual interpenetration.

We also considered the possibility of updating the hand mesh when a surface point of a finger slips. To do so, we selectively freeze fingers being in contact and independently apply the relative transformation of a moving surface point onto the corresponding finger. In this way, each finger is updated individually. However, depending on the translation, this may cause visually distorted hand meshes. To take into account the constraints between the fingers, one could use for example a spring-damper model [5], which limits the free motion for each individual phalanx. We will explore this direction in future work.

3.5 Implementation Aspects

We implemented our approach in Unity 5 with PhysX 3. We provide here several important implementation details, which are critical for efficient integration within the physics engine. These include finding and updating the contact points; applying the forces within the PhysiX engine, handling arbitrary, possibly non-convex, object shapes; and correctly handling the interpenetration of hand and object.

Contact Point Estimation. Our solution to efficiently identify accurate contact points uses the collision detection of Unity 5. Even if there is no real collision between the hand model and the virtual objects in practice, we are able to estimate contact points: We define a very small *Default Contact Offset*, a property of the physics engine, as a threshold to fire a collision event before the actual contact happens as explained in Section 3.2. The Default Contact Offset acts as a margin at which point the collision detection of Unity 5 fires collision events. We set the collision detection mode of Unity to *Continuous Dynamic* to support fast hand movements. From this collision event we receive a contact point and all involved colliders. We use this early contact point to manually cast a ray in the direction of the center of the corresponding sub-collider of the virtual object to get the actual surface point.

Surface Point Updates. As long as the contact remains in the static state, the contact points on the object's surface must not be updated. As soon as the state switches to dynamic, we update the corresponding contact point along the object's surface as illustrated in Fig. 3. From the previous contact point, we apply a transformation as a small offset in normal direction of the surface and cast a ray in direction of the interpenetrating phalanx position $\mathbf{p}_j$. This solution accounts for an engine limitation that does not allow for collision detection if a ray originates inside a collider. The ray cast ensures that we can estimate a new contact point directly on the object surface. Our solution detects occluding surface parts of the object by comparing the orientation of the surface normal vector of the original estimated contact point with the normal orientation of the newly calculated surface point. This ensures that a contact remains in a blocked area illustrated with the rays $\mathbf{R}_4$ and $\mathbf{R}_5$ in Fig. 3. Our proposed solution allows for efficient surface point updates without the need for additional *God-Objects* for every phalanx. Scene queries such as raycasts are very efficient in the Nvidia PhysiX engine and can be further accelerated by using *Volume-Caches* and *Single Object Caching*.

Applying Forces. Our approach acts as a middleware between the hand input and the Nvidia PhysiX physics engine. We compute the contact force of each phalanx for every frame and apply them via the *Application Programming Interface (API)* of the physics engine. More precisely, we specify the magnitude and direction of the force as well as the actual contact point on the object's surface where the force is applied. In our implementation, the external forces such as gravitational acceleration and collision forces between objects are handled by the physics engine internally.

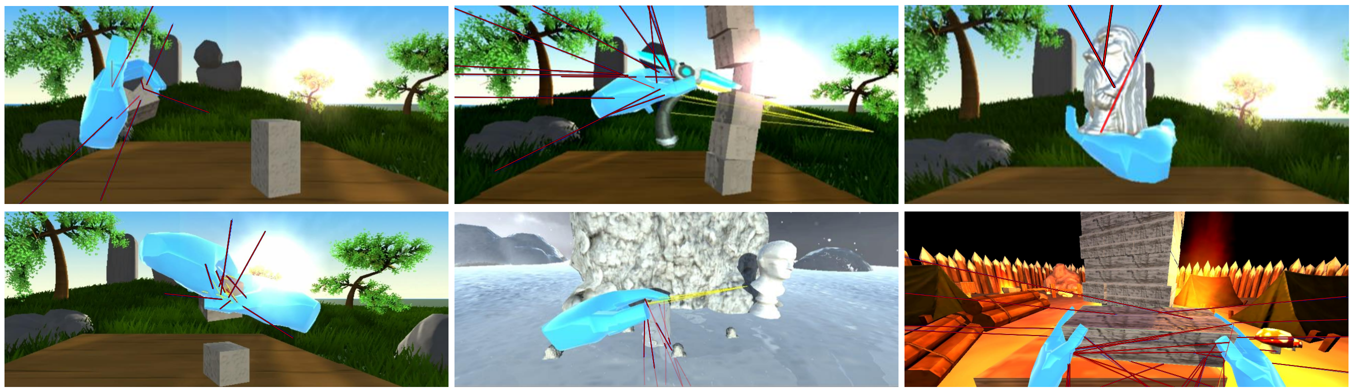


Figure 8: From left to right, top down: Palmar grasping, Grasping an object and use it as a tool on other objects, lifting a figurine using only the palm, Handover a virtual object midair to another hand, Sliding an object along a path, Using both hands to pull out a block from a tower.

Arbitrary Object Shapes. Physics engines have problems with complex, non-convex object meshes and only support convex objects or primitive shapes [10, 41]. For example, Nvidia PhysX 3, which we use in this work, does not support collisions between concave mesh colliders and dynamic non-kinematic rigid bodies. A common solution is to approximate a non-convex shape by its convex hull [41]. However, convex mesh approximations are too imprecise representations of the object mesh and this would significantly limit the interaction possibilities for fine-grained dexterous manipulation. Therefore, we approximate the concave objects with small voxels, by using a Finite Element Method [46]. The voxelization is more efficient than real concave mesh colliders and is easy to configure for different Levels of Detail.

Hand Model Interpenetration. To avoid interpenetration between the hand model and the virtual object, we disable the rigid body property of the corresponding phalanx once we have estimated a contact point. In Unity, physics colliders that are set as *trigger* do not cause real collision detection nor collision resolution. By setting the corresponding phalanges as triggers, we prevent the engine to generate any physical reactions between hand phalanges and the virtual objects, but we are still fully capable to track the current 3D transformation of the phalanges.

4 Results

We present here several interactions that demonstrate the capabilities of our approach. During all our tests, our approach could manage stable frame-rates without any performance issues. We added complex shaders, foliage, transparency, lighting, etc. to stress the graphics engine. Still, the test scenes run at over 60fps on an Intel i7 with 2.6GHz with an NVIDIA GTX 980M graphics card. We show different samples in Fig. 8. The illustrated interactions include: grasping an object using fingers and palm, grasping an axe to hit a tower of cubes that subsequently collapses, lifting a virtual object using only the palm, handover a virtual cube midair from one hand to the other, sliding a cylindrical head sculpture along a path depicted by small stones on the ground and finally pulling a block from a tower using both hands similar to the game *Jenga*.

5 Evaluation

In this section, we evaluate our approach for hand-object interaction. We conducted a pilot study among the users that participated to the evaluation process. In addition to that, we provide a performance analysis for the physics simulation with different configurations varying the number of contacts between the hand skeleton and virtual objects. We show that our method, while producing realistic object manipulations, is capable of real-time performance.

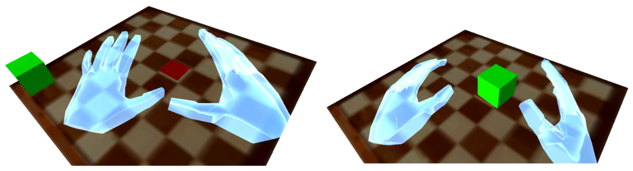


Figure 9: Scene used for evaluation. It shows a chessboard with a cube object, and a user is asked to move virtual objects (cubes, figurines, etc.) from a any position to a specified location on the chessboard shown in red.

5.1 Pilot Study

We created a simple VR environment as shown in Fig. 9 where the users could get familiar with each of the evaluated interaction methods and get a sense of its capabilities. We used the Oculus Rift DK2 to display the 3D scene to the users and the Leap Motion sensor mounted on the front plate. 12 different users participated in our evaluation. All the users were familiar with VR, and half of the users had hands-on experiences with VR.

We compare our approach to two other recent real-time approaches. The first baseline is a purely kinematic approach, which is our re-implementation of the kinematic part of [20]. In this approach, the object is considered as grasped when some measured forces reach a self defined threshold. Then the global transformation of the hand is applied to the object which is set to kinematic. We re-implemented the same kinematic motion control, while we simplified the prior grasp condition. As a second baseline, we use an industry method, the Leap Motion Interaction Engine 1.0[4], which is also a kinematic approach and very efficient for object placement due to its simplicity. We placed several different objects in the scene, however for the Interaction Engine we could only place simple objects due to its strict limitations.

After the users felt familiar with the interaction method, they were asked to move a virtual object to a specific target location on the chessboard. The target positions were randomized. This gave the users a chance to use each interaction system for a practical use case.

As we conducted the pilot study after some tries, the users were given several questions as shown in Table 1. Each question can be rated on a Likert Scale from 1 to 5, 5 being the best grade. 'Naturalness' refers to the question "How natural does the interaction feel?" and 'Diversity' refers to the question "Rate the diversity of interaction possibilities". The exact formulation of all the questions are given in the supplementary material.

[4]Publicly available at `https://developer.leapmotion.com/unity/`

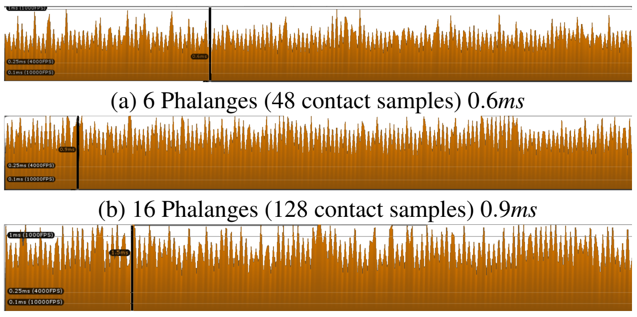
The Shapiro-Wilk test indicated that the normality assumption does not hold for our data. We therefore rely here on the non-parametric Friedman test to identify statistically significant differences in the data and report the corresponding chi-squared test statistics. When analyzing the results from Table 1, we obtain statically significant results for *Naturalness* ($\chi^2(2) = 16.68, p < 0.0005$), for *Usefulness* ($\chi^2(2) = 20.53, p < 0.0005$), and *Diversity* ($\chi^2(2) = 19.51, p < 0.0005$). The results for *Easy learning* are not statistically significant ($\chi^2(2) = 5.11, p = 0.078$).
Further, we apply the post hoc Wilcoxon Signed-Rank test on statistically significant data. The test indicated that the participants strongly agreed that our approach was perceived more natural to use (*Naturalness*) compared to the Interaction Engine ($Z = -2.689, p = 0.0035$) and [20] ($Z = -2.807, p = 0.0025$). Also, the participants mutually agreed that our approach allows more diverse interactions (*Diversity*) than the Interaction Engine ($Z = -2.727, p = 0.0032$) and [20] ($Z - 2.824, p = 0.0024$). The ratings for usefulness were also significantly higher for our method than for the Interaction Engine ($Z = -2.421, p = 0.0077$) and [20] ($Z = -2.712, p = 0.0034$).

5.2 Performance Analysis

We evaluate our method in terms of real-time performance, since computational efficiency is one of the most critical aspects for VR integration. The frame rate analysis, presented in Fig. 10, has been performed with the same hardware setup as reported in Section 4. Each image shows the number of phalanges being in contact with an object, the number of sampled contact points exerting forces (8 samples per contact point), and the computation time over multiple frames. We state the total computational cost of the physics simulation, which includes all physics simulations of the scene, as well as the simulation of our interaction model. The simulation is very fast, taking only $0.6ms$ for 6 phalanges, or $1.5ms$ for 32 phalanges, which is significantly faster than the timings reported by Talvas *et al.* [39], who use a more complex friction model. The evaluation clearly shows the efficiency of our approach, which is essential for real-time applications.

5.3 Discussion

The users responded very positively to our physics-based method since there are many interaction possibilities and it feels more realistic and natural. Our experimental evaluations showed that users preferred to use friction to let the object slide between fingers and the chessboard. They tried to use that interaction also during the evaluations of the other two methods, but those approaches do not support that, at least not in a similar way. Users responded less positively to the other two methods because these methods felt very artificial and limited since there is no friction contact possible. While the users rated the naturalness, usefulness, and diversity of our method much higher than for the other methods, they stated that the other methods were easier to learn. However, the conducted statistical tests indicate the *Easy learning* criteria as not statistically significant. Overall, 8 out of 12 users preferred our physics-based method over the other two methods. We conclude that our method is more realistic as it allows for more diverse interactions. Despite the simplicity of the Coulomb friction model, our approach produces realistic object manipulations and proves to be capable of achieving real-time performance for VR applications.



(a) 6 Phalanges (48 contact samples) $0.6ms$

(b) 16 Phalanges (128 contact samples) $0.9ms$

(c) 32 Phalanges (256 contact samples) $1.5ms$

Figure 10: Performance analysis of the physics computation. The graph illustrated the run-time of the physics simulation with varying number of phalanges being in contact. Each contact is sampled with 8 contact samples within the patch area. The black bar depicts one frame in the timeline for which we state the simulation time.

Table 1: Results of our pilot study. We asked the users to answer several questions about the interaction method and rate them on a Likert Scale from 1 to 5, with 5 being the favourable value. We report median and inter-quartile range. Bold results indicate statistical significance.

Input method → / User question ↓	Ours	Re-impl. [20]	Interaction Engine
Naturalness	**4**(**1**)	2(0.75)	2.5(2)
Usefulness	**4.5**(**1**)	2(0)	3(1)
Diversity	**4.5**(**1**)	2(1.75)	3(1)
Easy learning	3(1)	4(1.75)	4.5(1)
Preference	**8**	1	3

6 Conclusion and Future Work

In this work we presented an efficient real-time approach for unconstrained hand object interaction in Virtual Reality environments. We proposed to rely on the Coulomb model to introduce friction, we showed how to efficiently implement it, and through our experiments and evaluation, we showed that it results in a natural way to interact with virtual objects. However, the Coulomb model is still a simple approximation, we consider to integrate more complex friction models, which need to be carefully selected with respect to the computational performance. Another interesting direction to explore are real-time simulations of soft-bodies, for example through particle-based simulation [23]. We believe that our work will provide a solid basis for the development of unconstrained hand object interaction in VR environments.

Acknowledgments

The authors wish to thank Christian Pirchheim for valuable discussions, and the participants of the user study for their time and feedback.

References

[1] L. Avila and M. Bailey. Augment your reality. *Computer Graphics and Applications*, 36(1):6–7, Jan.–Feb. 2016.

[2] C. Basdogan, C.-H. Ho, and M. Srinivasan. Ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments. *American Society of Mechanical Engineers, Dynamic Systems and Control Division*, 61:77–84, Jan. 1997.

[3] H. Benko, R. Jota, and A. Wilson. MirageTable: Freehand interaction on a projected augmented reality tabletop. In *Proc. CHI*, pp. 199–208. ACM, 2012.

[4] C. W. Borst and A. P. Indugula. Realistic virtual grasping. In *Proc. VR*, pp. 91–98. IEEE, 2005.

[5] C. W. Borst and A. P. Indugula. A spring model for whole-hand virtual grasping. *Presence: Teleoperators and Virtual Environments*, 15(1):47–61, Feb. 2006.

[6] A. Bracegirdle, T. Mitrovic, and M. Mathews. Investigating the usability of the leap motion controller: Gesture-based interaction with a 3D

virtual environment. Technical report, University of Canterbury, NZ, 2014.

[7] V. Buchmann, S. Violich, M. Billinghurst, and A. Cockburn. FingARtips: gesture based direct manipulation in augmented reality. In *Proc. GRAPHITE*, pp. 212–221. ACM, 2004.

[8] W.-H. Chen, D. J. Ballance, P. J. Gawthrop, and J. O'Reilly. A nonlinear disturbance observer for robotic manipulators. *IEEE Transactions on Industrial Electronics*, 47(4):932–938, Aug. 2000.

[9] J. Egger, M. Gall, J. Wallner, P. Boechat, A. Hann, X. Li, X. Chen, and D. Schmalstieg. Integration of the HTC Vive into the medical platform MeVisLab. In *Proc. SPIE*, pp. 10138 – 10138 – 7. SPIE, 2017.

[10] T. Erez, Y. Tassa, and E. Todorov. Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX. In *Proc. ICRA*, pp. 4397–4404. IEEE, 2015.

[11] M. Gafvert. Comparisons of two dynamic friction models. In *Proc. ICCA*, pp. 386–391. IEEE, 1997.

[12] C. Garre, F. Hernandez, A. Gracia, and M. A. Otaduy. Interactive simulation of a deformable hand for haptic rendering. In *Proc. WHC*, pp. 239–244. IEEE, 2011.

[13] J.-P. Gourret, N. M. Thalmann, and D. Thalmann. Simulation of object and human skin formations in a grasping task. *ACM SIGGRAPH*, 23(3):21–30, July 1989.

[14] J. Guna, G. Jakus, M. Pogačnik, S. Tomažič, and J. Sodnik. An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors*, 14(2):3702–3720, Feb. 2014.

[15] C. Hand. A survey of 3D interaction techniques. *Computer Graphics Forum*, 16(5):269–281, Dec. 1997.

[16] O. Hilliges, D. Kim, S. Izadi, M. Weiss, and A. Wilson. HoloDesk: direct 3D interactions with a situated see-through display. In *Proc. CHI*, pp. 2421–2430. ACM, 2012.

[17] J. Jacobs and B. Froehlich. A soft hand model for physically-based manipulation of virtual objects. In *Proc. VR*, pp. 11–18. IEEE, 2011.

[18] J. Jacobs, M. Stengel, and B. Froehlich. A generalized god-object method for plausible finger-based interactions in virtual environments. In *Proc. 3DUI*, pp. 43–51. IEEE, 2012.

[19] R. Kelly, J. Llamas, and R. Campa. A measurement procedure for viscous and coulomb friction. *IEEE Transactions on Instrumentation and Measurement*, 49(4):857–861, Aug. 2000.

[20] J. Kim and J. Park. Physics-based hand interaction with virtual objects. In *Proc. ICRA*, pp. 3814–3819. IEEE, 2015.

[21] J.-S. Kim and J.-M. Park. Direct and realistic handover of a virtual object. In *Proc. IROS*, pp. 994–999. IEEE/RSJ, 2016.

[22] Y. Li, J. L. Fu, and N. S. Pollard. Data-driven grasp synthesis using shape matching and task-based pruning. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):732–747, July–Aug. 2007.

[23] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics*, 33(4):153:1–153:12, July 2014.

[24] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. In *Proc. ICRA*, pp. 1824–1829. IEEE, 2003.

[25] M. Moehring and B. Froehlich. Effective manipulation of virtual objects within arm's reach. In *Proc. VR*, pp. 131–138. IEEE, 2011.

[26] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural network. In *Proc. CVPR*, pp. 4207–4215. IEEE, 2016.

[27] K. Nasim and Y. J. Kim. Physics-based interactive virtual grasping. In *Proc. HCIK*, pp. 114–120. Hanbit Media, Inc., 2016.

[28] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands deep in deep learning for hand pose estimation. In *Proc. CVWW*, pp. 21–30. Graz University of Technology, 2015.

[29] H. Olsson, K. J. Åström, C. C. De Wit, M. Gäfvert, and P. Lischinsky. Friction models and friction compensation. *European journal of control*, 4(3):176–195, Mar. 1998.

[30] M. Ortega, S. Redon, and S. Coquillart. A six degree-of-freedom god-object method for haptic display of rigid bodies with surface properties. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):458–469, May–June 2007.

[31] V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, July 1997.

[32] A. G. Perez, G. Cirio, F. Hernandez, C. Garre, and M. A. Otaduy. Strain limiting for soft finger contact simulation. In *Proc. WHC*, pp. 79–84. IEEE, 2013.

[33] N. S. Pollard and V. B. Zordan. Physically based grasping control from example. In *Proc. SCA*, pp. 311–318. ACM, 2005.

[34] M. Prachyabrued and C. W. Borst. Virtual grasp release method and evaluation. *International Journal of Human-Computer Studies*, 70(11):828–848, Nov. 2012.

[35] M. Prachyabrued and C. W. Borst. Visual feedback for virtual grasping. In *Proc. 3DUI*, pp. 19–26. IEEE, 2014.

[36] S. S. Rautaray and A. Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Journal of Artificial Intelligence Research*, 43(1):1–54, Jan. 2015.

[37] H. Rijpkema and M. Girard. Computer animation of knowledge-based human grasping. *SIGGRAPH Computer Graphics*, 25(4):339–348, July 1991.

[38] D. J. Sturman, D. Zeltzer, and S. Pieper. Hands-on interaction with virtual environments. In *Proc. UIST*, pp. 19–24. ACM, 1989.

[39] A. Talvas, M. Marchal, C. Duriez, and M. A. Otaduy. Aggregate constraints for virtual manipulation with soft fingers. *IEEE Transactions on Visualization and Computer Graphics*, 21(4):452–461, Apr. 2015.

[40] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, et al. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics*, 35(4):143, July 2016.

[41] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Proc. IROS*, pp. 5026–5033. IEEE/RSJ, 2012.

[42] C. Von Hardenberg and F. Bérard. Bare-hand human-computer interaction. In *Proc. PUI*, pp. 1–8. ACM, 2001.

[43] L. Wei and A. Sourin. Function-based approach to mixed haptic effects rendering. *The Visual Computer*, 27(4):321–332, Apr. 2011.

[44] C. A. Wingrave, B. Williamson, P. D. Varcholik, J. Rose, A. Miller, E. Charbonneau, J. Bott, and J. J. LaViola Jr. The wiimote and beyond: Spatially convenient devices for 3D user interfaces. *Computer Graphics and Applications*, 30(2):71–85, Mar.–Apr. 2010.

[45] D. Yim, G. N. Loison, F. H. Fard, E. Chan, A. McAllister, and F. Maurer. Gesture-driven interactions on a virtual hologram in mixed reality. In *Proc. ISS Companion*, pp. 55–61. ACM, 2016.

[46] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, S. S. Samant, and J. E. Bussoletti. A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics. *Journal of Computational Physics*, 92(1):1–66, jan 1991.

[47] E. Zell, C. Aliaga, A. Jarabo, K. Zibrek, D. Gutierrez, R. McDonnell, and M. Botsch. To stylize or not to stylize?: The effect of shape and material stylization on the perception of computer-generated faces. *ACM Transactions on Graphics*, 34(6):184:1–184:12, Nov. 2015.

[48] M. Zhang and A. Mak. In vivo friction properties of human skin. *Prosthetics and Orthotics International*, 23(2):135–141, Aug. 1999.

[49] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE Multimedia*, 19(2):4–10, Feb. 2012.

[50] W. Zhao, J. Zhang, J. Min, and J. Chai. Robust realtime physics-based motion control for human grasping. *ACM Transactions on Graphics*, 32(6):207, Nov. 2013.

[51] C. Zimmermann and T. Brox. Learning to estimate 3D and pose from single RGB images. In *Proc. ICCV*, pp. 4903–4911. IEEE, 2017.